# VERİ SIKIŞTIRMA

# Hata Bulma Ve Düzeltme-2

- Veri iletilirken bazı bitlerde bozulmalar olabilir. Bu bozulmalara **hata** denir.

- Ortama bağlı olarak bu hataların oluşma olasılığı değişiklik gösterir.
- Atmosferik olaylar (yıldırım, vs.) telli ve telsiz ortamlarda hataya sebep olurlar.
- Meydana gelen bozulmalar çoğu uygulama tarafından kabul edilmez.

- Veri içerisindeki 1 bitlik bozulma tüm verinin yanlış anlaşılmasına neden olabilir. Bu sebeple iletişim sırasında veri içerisinde bir bozulmanın olup olmadığının anlaşılması gerekir.

- Bu sebeple hata sezme ve düzeltme teknikleri kullanılır. Hatayı sezme işlemi hatayı düzeltmeye göre daha kolaydır.
- Bazen hatayı sezebilir ama düzeltemeyiz.

- Bir bitlik bozulmaları sezmek ve düzeltmek kolaydır, ancak bir bit üzeri bozulmaları sezsek bile düzeltemeyiz, verinin yeniden gönderilmesi gerekir.

Veri iletiminde iki çeşit hata söz konusudur.

**1-Patlama hatası (burst error):** Çevre koşulları nedeniyle alıcıya gerçek olmayan anlamsız bilgiler gelir. (yıldırım v.s 1-100 ms)

**2-Rasgele hata (random error):** İletim yolundaki elektriksel gürültü nedeniyle bilgi bloğu içinde rastgele bir bitin bozulması söz konusudur.

**İletim hatalarının veriye olan etkisi**

- Çok kısa süren girişim, **spike** olarak adlandırılır.
Tek bitlik hataya sebep olur.

- Uzun süren girişim veya bozulma **burst** (birden fazla bitlik) hatalara neden olur.
Sinyal açık olarak 1 veya 0 değildir. Öyle belirsiz bir alana düşer ki bu **erasure** (silinmiş yer) olarak adlandırılır.

## En çok kullanılan hata denetim metotları

### 1- FEC (Error Correction Code - Forward Error Correction-İleri Yönlü Hata Denetimi)

- Hata denetimi için gönderilen veri kümesine ek bitler ekler
- Hatayı bulur ve gerekirse alıcıda düzeltmeye çalışır
- Hız kaybı yaşanmaz
- Çok gürültülü ortamlarda kullanılamaz
- FEC metodu, yeniden iletimin zor veya imkansız olduğu bağlantılarda ve veri kümesinin küçük olduğu uygulamalarda kullanılır
- Katlamalı kodlar, BCH kodlar, **Hamming Kodlar**, Reed-Salamon kodları

### 2- ARQ (Automatic Repeat Request- Otomatik Tekrar İsteği)

- Hatayı bulma ve bozulan verinin yeniden iletilmesi için alıcı taraftan istekte bulunulması işlemidir
- Uygulamalarda bu teknik kullanılır, çünkü aynı hatayı tespit için gerekli bit sayısı, düzeltmek için kullanılan bit sayısından çok daha azdır.
- **CRC kodları**, Seri Eşlik (Parity), Blok Eşlik, Modül Toplamı

## Uygulama

**Gönderilen Veri :** 1 0 1 **1** 0 1 0 0 1 1 0 ……

**Alınan Veri** : 1 0 1 **0** 0 1 0 0 1 1 0 …… Hata sezilemez

### Her bir biti çift sayıda yazalım

**Gönderilen Veri :** 11 00 11 **11** 00 11 00 00 11 11 00 ……

**Alınan Veri** : 11 00 11 **01** 00 11 00 00 11 11 00……

Hatayı sezeriz ama düzeltemeyiz

### Her bir biti iki kopya ile yazalım

**Gönderilen Veri :** 111 000 111 **111** 000 111 000 000 111 111 000 ……

**Alınan Veri** : 111 000 111 **011** 000 111 000 000 111 111 010……

Hatayı sezeriz bir bitlik bozulma var ise düzeltebiliriz aksi durumda düzeltemeyiz

**Her iki bit için 5 bitlik bir kod kelimesi kullanırsak**

00 → 00001
01 → 01010
10 → 10100
11 → 11111

Kod kelimeleri seçilirken en az 3 bitte değişiklik olacak şekilde seçilmiştir.

**Orijinal mesaj**

10 11 01 00 11… olsun

10100 11111 01010 00001 11111 Bir bitte oluşan bozulmalar kesinlikle düzeltilebilir

| Bozulmamış Kod kelime | 00001 | 01010 | 10100 | 11111 |
|---|---|---|---|---|
| Mümkün olan bir bitteki bozulma | 10001 | 11010 | 00100 | 01111 |
| | 01001 | 00010 | 11100 | 10111 |
| | 00101 | 01110 | 10000 | 11011 |
| | 00011 | 01000 | 10110 | 11101 |
| | 00000 | 01011 | 10101 | 11110 |

# Hamming Kodlama

4 bitlik verideki yer değiştirmeler ile 7 bitlik kod-kelimelerinin elde edildiği lineer bir kodlamadır. (1 bit ≈1,75 bit)

- Bütün kod kelimelerinde en az 3 bitte değişim vardır.
- Bir bitte oluşan bozulmaları düzeltebiliriz.

⊕ parity addition işlemidir. (01/10)→1 (00/11)→ 0

| mesaj | | Kod-kelimesi |
|---|---|---|
| 0000 | | 0000000 |
| 0001 | | 0001011 |
| 0010 | | 0010111 |
| 0100 | | 0100101 |
| 1000 | | 1000110 |
| 1100 | | 1100011 |
| 1010 | | 1010001 |
| 1001 | | 1001101 |
| 0110 | | 0110010 |
| 0101 | | 0101110 |
| 0011 | | 0011100 |
| 1110 | | 1110100 |
| 1101 | | 1101000 |
| 1011 | | 1011010 |
| 0111 | | 0111001 |

$b_1b_2b_3b_4\ (b_{1\oplus}b_{2\oplus}b_3)(b_{1\oplus}b_{3\oplus}b_4)(b_{2\oplus}b_{3\oplus}b_4)$

Hamming Kodlamada 3 şart geçerlidir

$b_1 \oplus b_2 \oplus b_3 \oplus b_5 = 0$
$b_1 \oplus b_3 \oplus b_4 \oplus b_6 = 0$
$b_2 \oplus b_3 \oplus b_4 \oplus b_7 = 0$

Kod çözme işleminde basit bir algoritma kullanılır.
Bozulmuş kod kelimesinin $\widetilde{b_1}\widetilde{b_2}\widetilde{b_3}\widetilde{b_4}\widetilde{b_5}\widetilde{b_6}\widetilde{b_7}$ olduğunu düşünelim ve 3 parity-check bilgisini kontrol edelim.

**Parity check kontrolü**

$P_1 = \widetilde{b_1} \oplus \widetilde{b_2} \oplus \widetilde{b_3} \oplus \widetilde{b_5}$
$P_2 = \widetilde{b_1} \oplus \widetilde{b_3} \oplus \widetilde{b_4} \oplus \widetilde{b_6}$
$P_3 = \widetilde{b_2} \oplus \widetilde{b_3} \oplus \widetilde{b_4} \oplus \widetilde{b_7}$

Hesaplanan $P_1$, $P_2$ ve $P_3$ farklı değerler aldığı görülmektedir. Hatayı düzeltmek için yapılması gerekenler.

- $P_1$=$P_2$=$P_3$=0 ise $\widetilde{b_1}\widetilde{b_2}\widetilde{b_3}\widetilde{b_4}\widetilde{b_5}\widetilde{b_6}\widetilde{b_7}$ doğru kod kelimesidir. Dördüncü bitten sağa doğru giderek $\widetilde{b_5}\widetilde{b_6}\widetilde{b_7}$ atılır.

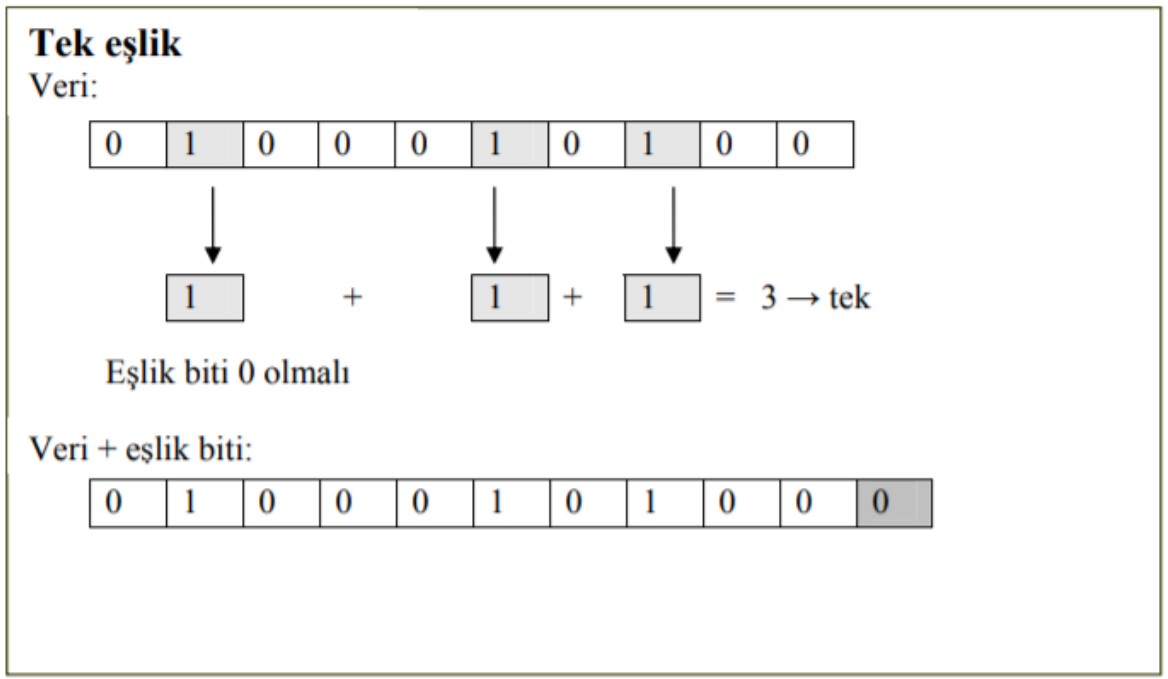
- Eğer $P_j$ lerden biri 1, diğer ikisi 0 ise; $\widetilde{b_l}$ nin hangi $P_j$ den geldiği bulunur. İlgili $P_j$ de olup, diğerlerinde olmayan $\widetilde{b_l}$ seçilir ve tersi alınır.
  Örnek
  $P_2$=1 $P_1$=$P_3$=0 ise $P_2$ de olup, $P_1$ ve $P_3$ de olmayan $\widetilde{b_6}$ bitinin tersi alınarak doğru kod kelimesi elde edilir.

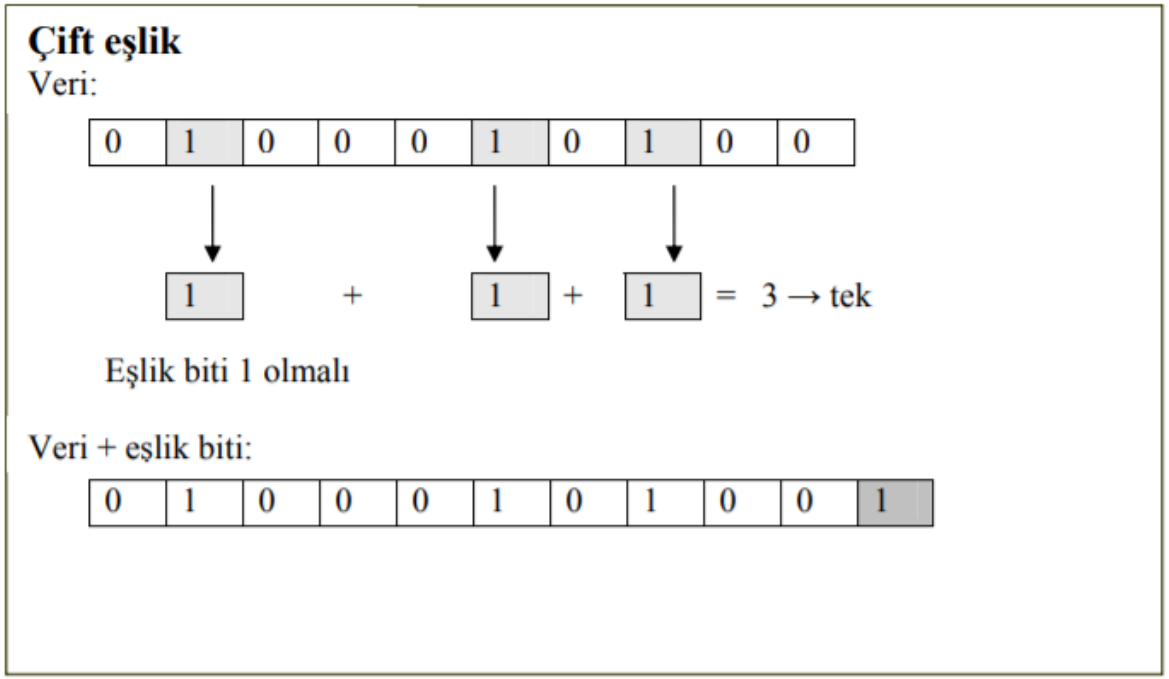
- Eğer $P_j$ lerden biri 0, diğer ikisi 1 ise; diğer ikisinde mevcut olup, $P_j$ de olmayan $\widetilde{b_l}$ bulunup, tersi alınır.
  Örnek
  $P_3$=0 $P_1$=$P_2$=1 ise $P_3$ de olup, $P_1$ ve $P_2$ de olmayan $\widetilde{b_1}$ bitinin tersi alınarak doğru kod kelimesi elde edilir.

- $P_1$=$P_2$=$P_3$=1 ise her üçünde olan $\widetilde{b_l}$ ($\widetilde{b_3}$) bitinin tersi alınarak doğru kod kelimesi elde edilir.

# Eşlik Sınaması

- Gönderilen veride oluşan tek sayıdaki hatayı sezmek için kullanılır.
- Verideki birlerin sayısını tek veya çift olarak düzenlemektir.
- Eşlik biti 1 veya 0 yapılarak tüm veri grubunun içindeki birlerin sayısının çift veya tek olması sağlanır.
- **Tek Eşlik yönteminde veri içerisindeki 1'lerin sayısı tek ise eşlik biti 0 olur.**
- **Çift Eşlik yönteminde veri içerisindeki 1'lerin sayısı tek ise eşlik biti 1 olur.**

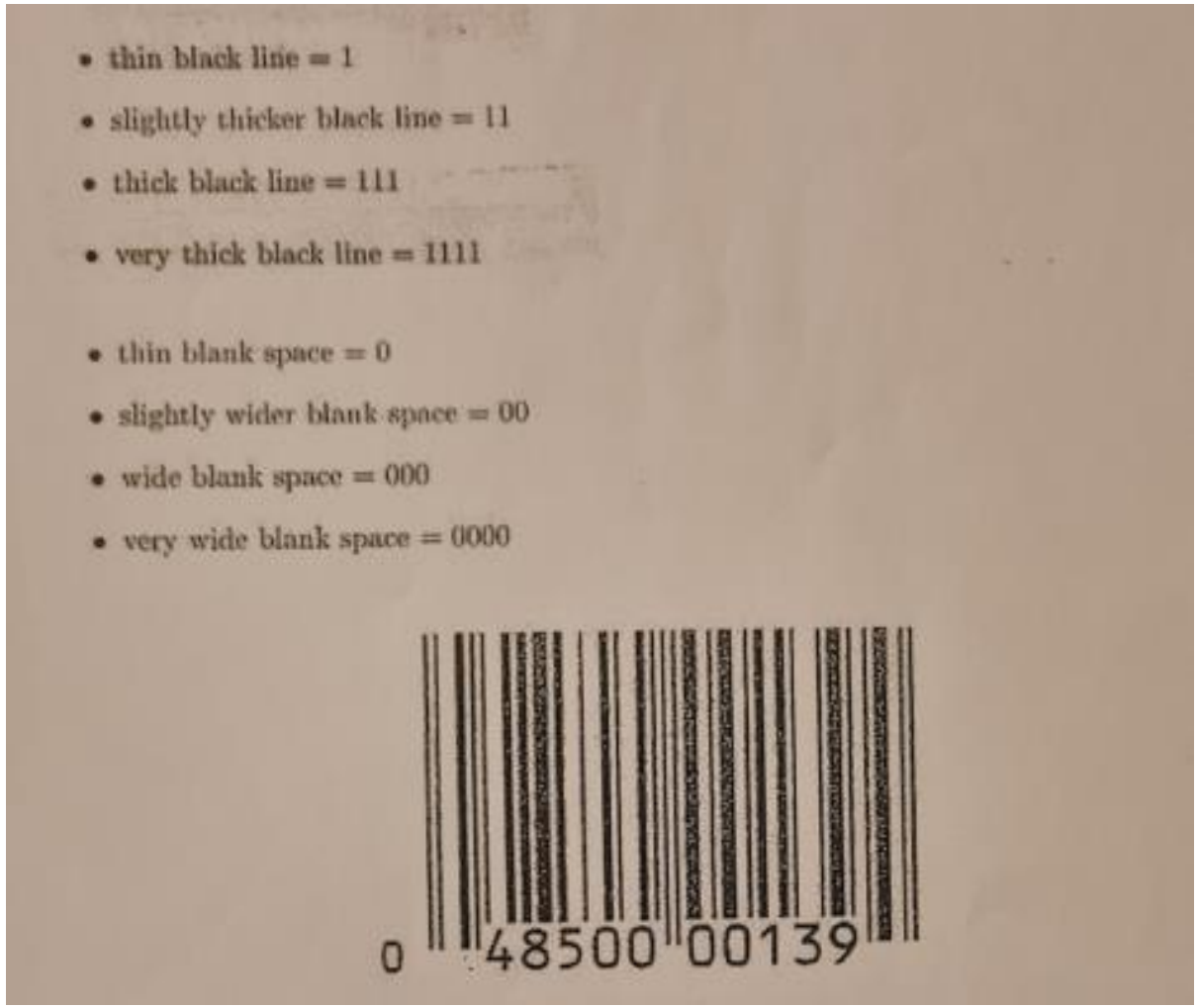



- 1, 3, 5… gibi tek sayıda hatayı sezer. Genellikle yedi, sekiz bit gibi kısa verilerin aktarımında kullanılır.

# Universal Product Code (UPC)

- Kod üzerinde bar kodun kendisi ve dijitleri olmak üzere iki farklı kısım yer alır.
- Her decimal dijit, 7 binary dijit ile kodlanır
- Kodun sol, sağ yanlarında ve ortasında separatörler vardır.




| | left | right |
|---|---|---|
| 0 | 0001101 | 1110010 |
| 1 | 0011001 | 1100110 |
| 2 | 0010011 | 1101100 |
| 3 | 0111101 | 1000010 |
| 4 | 0100011 | 1011100 |
| 5 | 0110001 | 1001110 |
| 6 | 0101111 | 1010000 |
| 7 | 0111011 | 1000100 |
| 8 | 0110111 | 1001000 |
| 9 | 0001011 | 1110100. |

Yukarıda bahsettiğimiz ürünün barkodu 048500 00139 4 olsun.
Bu sayı UPC bar code ile 6+6 dijit olarak kodlanmıştır.

1 + 5 + 5 + 1 dijit

- 1 → ürünün ne tip ürün olduğu ?
- 5 → üreticinin kodu
- 5 → ürünün kodu
- 1 → kontrol dijiti ( Barcode da olmasına rağmen decimal sayı değeri rakkamıle yazılmaz)

Notlar:

* Sol taraftaki her 7'li grup 0 ile başlar 1 ile biter.
  Sağ taraftaki her 7'li grup 1 ile başlar 0 ile biter.
* Sol taraftaki her 7'li grup içerisinde birlerin sayısı tek tir.
  Sağ taraftaki her 7'li grup içerisinde birlerin sayısı çift tir.

| | left | right |
|---|---|---|
| 0 | 0001101 | 1110010 |
| 1 | 0011001 | 1100110 |
| 2 | 0010011 | 1101100 |
| 3 | 0111101 | 1000010 |
| 4 | 0100011 | 1011100 |
| 5 | 0110001 | 1001110 |
| 6 | 0101111 | 1010000 |
| 7 | 0111011 | 1000100 |
| 8 | 0110111 | 1001000 |
| 9 | 0001011 | 1110100. |

## Kontrol dijit

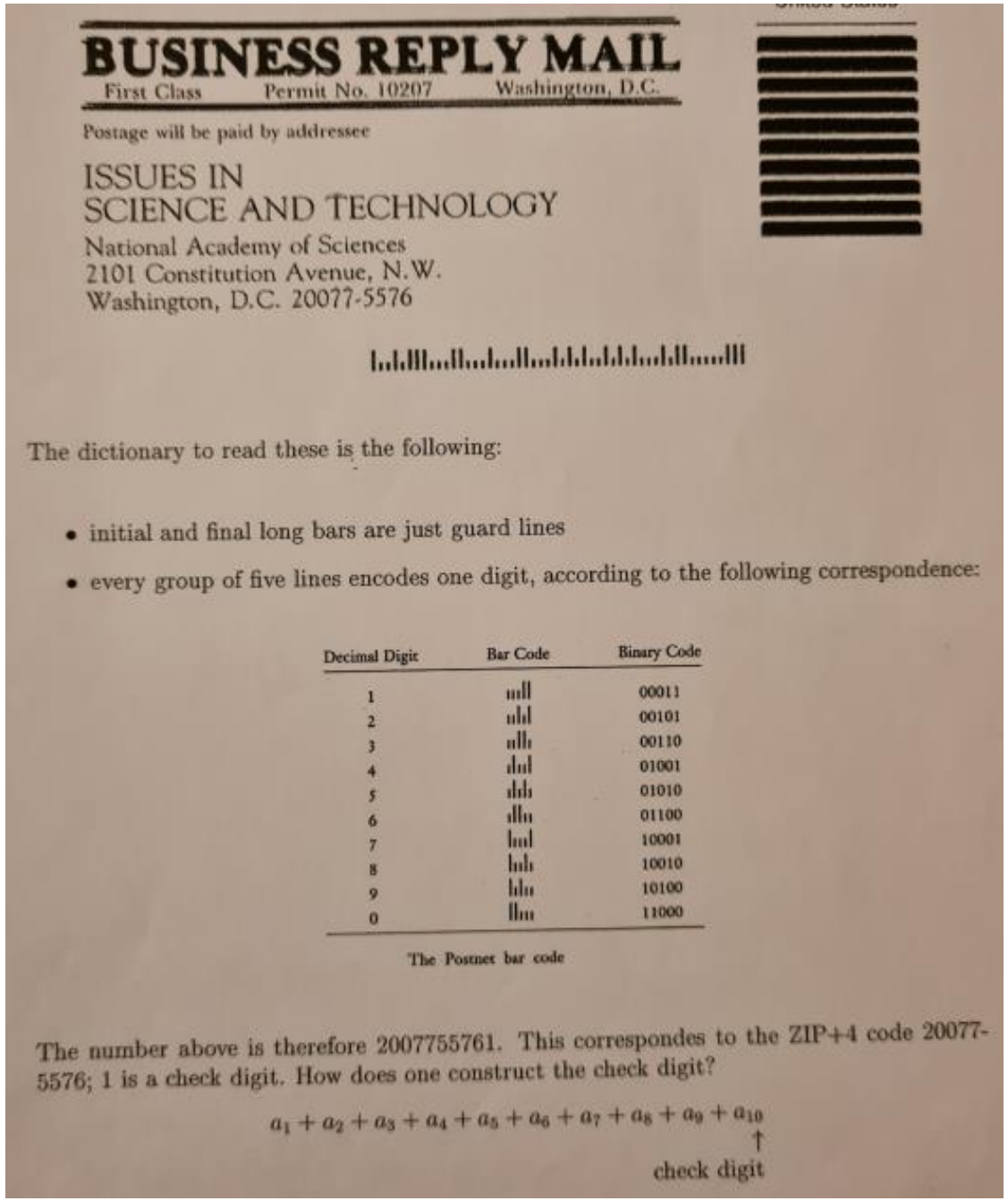
ilk 11 dijiti alıp hesaplayalım

$$3a_1 + a_2 + 3a_3 + a_4 + 3a_5 + a_6 + 3a_7 + a_8 + 3a_9 + a_{10} + 3a_{11} \equiv A \pmod{10}$$

Kontrol dijiti C veya $a_{12}$ ise $a_{12} = C = 10 - A \pmod{10}$.

$$3a_1 + a_2 + 3a_3 + a_4 + 3a_5 + a_6 + 3a_7 + a_8 + 3a_9 + a_{10} + 3a_{11}$$

$$4 + 24 + 5 + 3 + 3 + 27 = 66 \Rightarrow A = 6 \Rightarrow a_{12} = C = 4$$



**BUSINESS REPLY MAIL**
First Class Permit No. 10207 Washington, D.C.

Postage will be paid by addressee

ISSUES IN
SCIENCE AND TECHNOLOGY
National Academy of Sciences
2101 Constitution Avenue, N.W.
Washington, D.C. 20077-5576

The dictionary to read these is the following:

- initial and final long bars are just guard lines
- every group of five lines encodes one digit, according to the following correspondence:

| Decimal Digit | Bar Code | Binary Code |
|---|---|---|
| 1 | ıılı | 00011 |
| 2 | ılıl | 00101 |
| 3 | ılıı | 00110 |
| 4 | ıllı | 01001 |
| 5 | ıllı | 01010 |
| 6 | ılıı | 01100 |
| 7 | lııl | 10001 |
| 8 | lılı | 10010 |
| 9 | lıll | 10100 |
| 0 | llıı | 11000 |

The Postnet bar code

The number above is therefore 2007755761. This correspondes to the ZIP+4 code 20077-5576; 1 is a check digit. How does one construct the check digit?

$$a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9 + a_{10}$$
$$\uparrow$$
check digit

Her grup 2 uzun, 3 kısa çubuktan oluşur. Kısa yerine uzun ve tam tersi durum olduğunda hata oluşur.

Oluşan bir bitlik hata düzeltilebilir mi ?

2007 ? 55761 Beşinci dijit hatalı olsun.

2+0+0+7+?+5+5+7+6+**1** = (33+?) mod 10

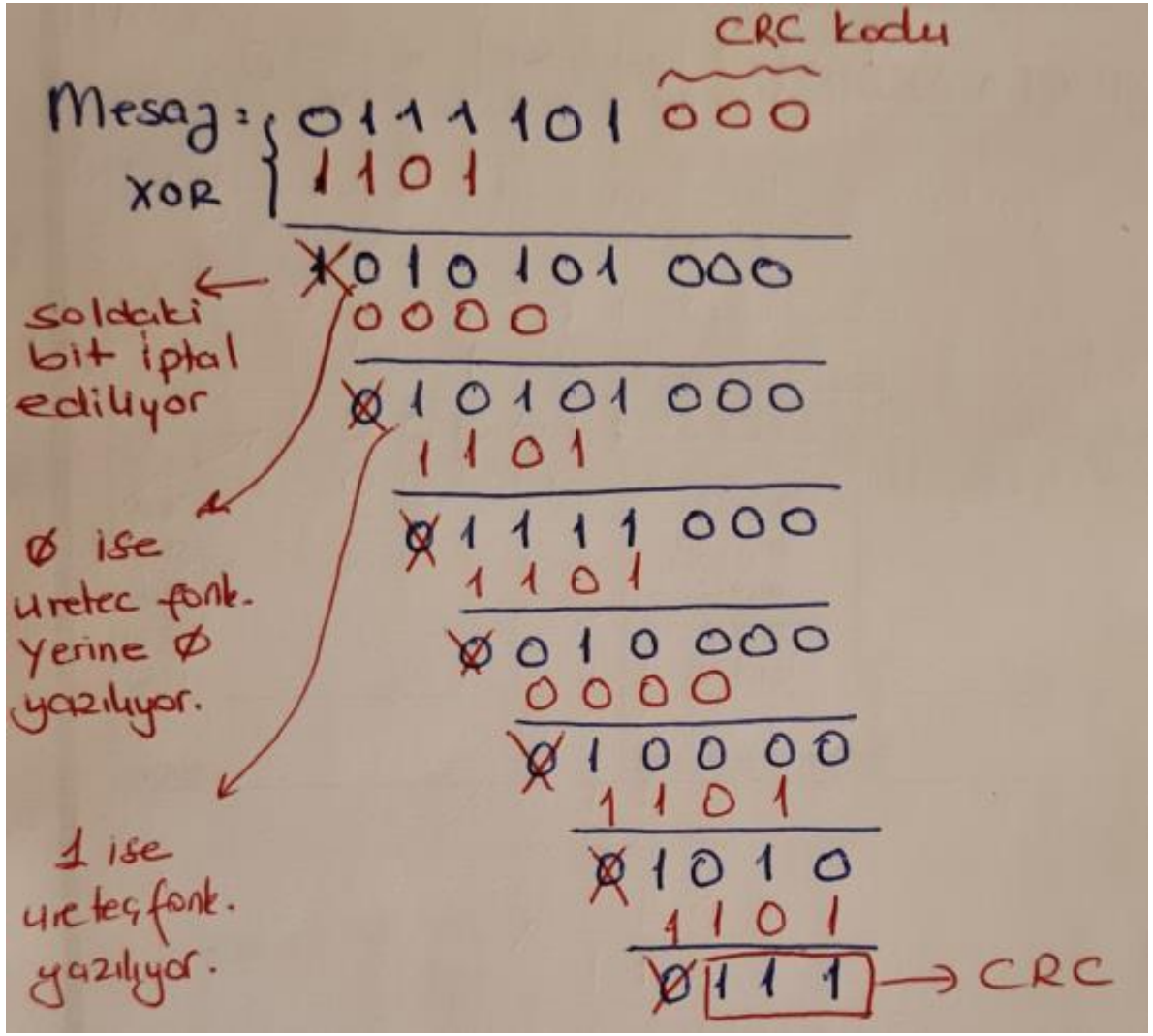
Kontrol dijit nasıl bulunur ?

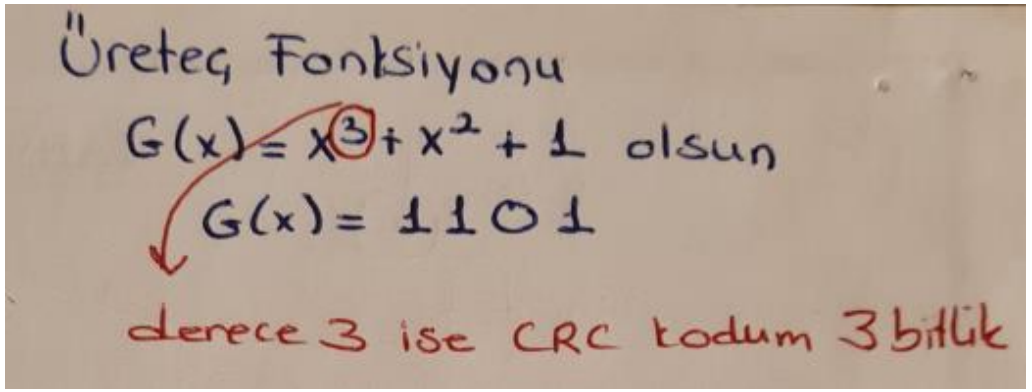
Dijitler toplamı 10'un katı olmalıdır.

2+0+0+7+7+5+5+7+6+**1** = (40 mod 10)=0

# CRC (Cyclic Redundancy Check)

- Gönderilecek mesaj **0111101**



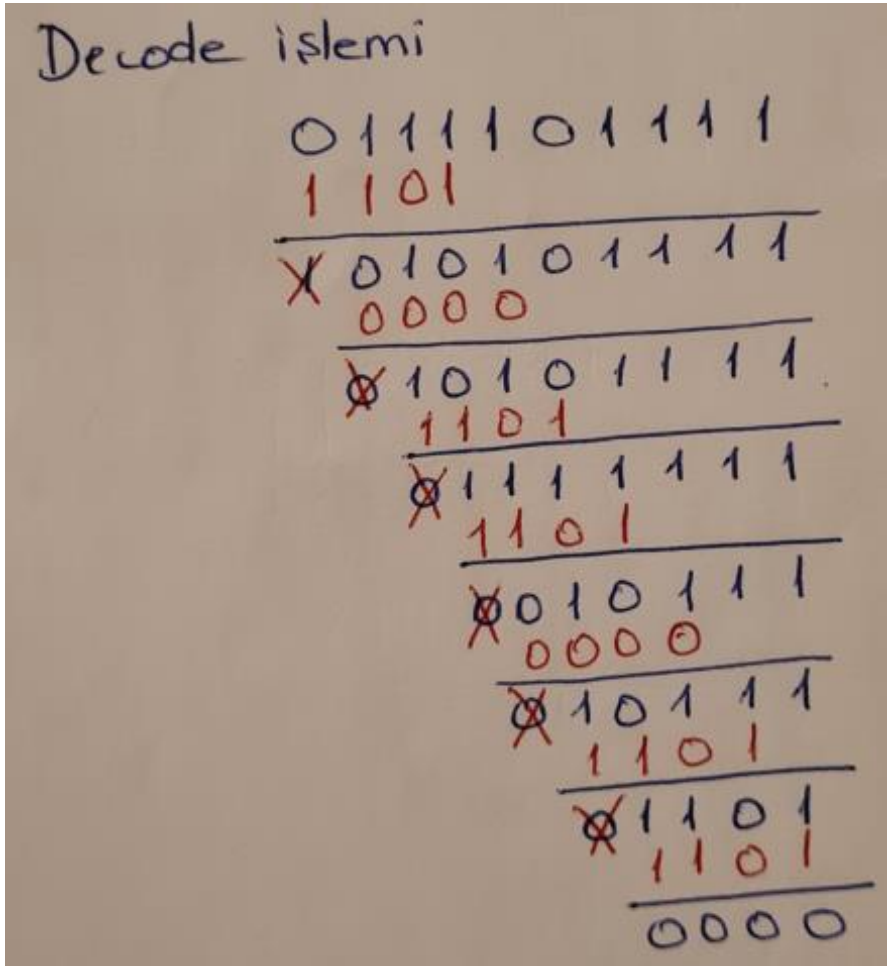

- G(x) : Üreteç fonksiyonumuz
- Üreteç fonksiyonunun derecesi CRC kodunun bit uzunluğunu verir ve başlangıçta değeri 0 dır.

- Encode edilmiş mesaj

**0111101111**

- Encode edilmiş mesaj **0111101111**

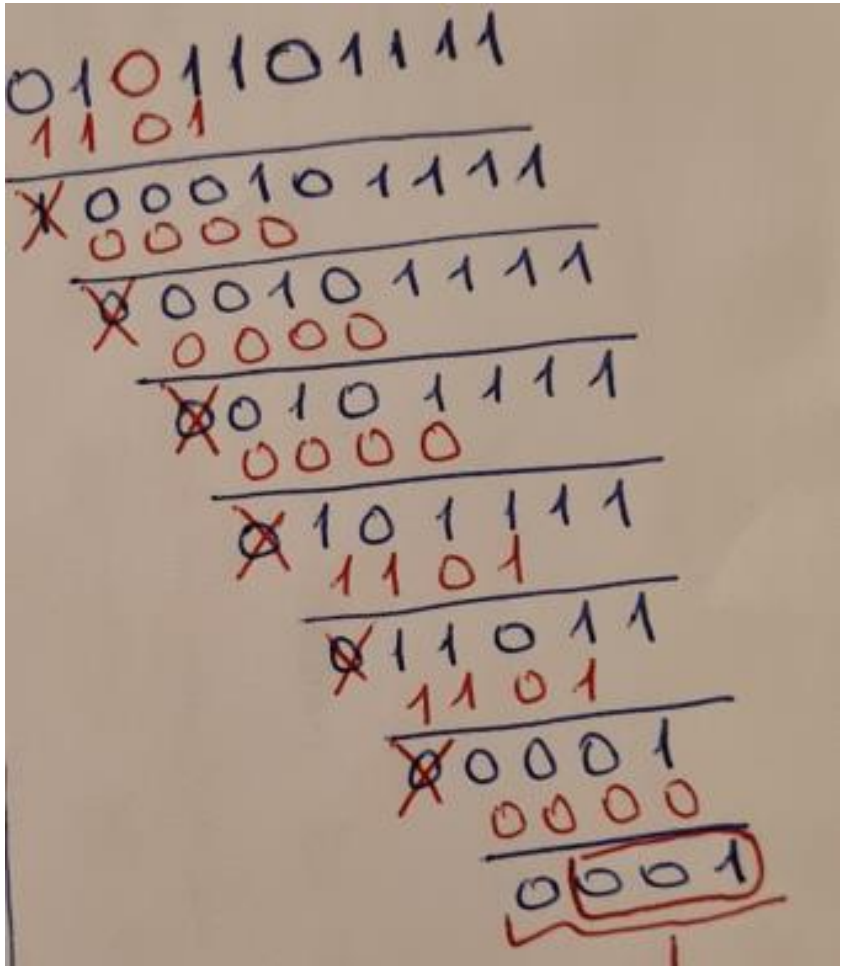
**Hatalı mı gitti ? Yoksa doğru mu?**




Bozulma YOK

- Encode edilmiş mesaj **0111101111**

**Üçüncü bit hatalı alınmış olsaydı, anlaşılacak mıyı?**




Bozulma VAR

# MD5 (Message Digest)

Bir internet standartı olup, veri güvenliğinde de kullanılır

- Veri güvenliği ve bütünlüğünde kullanılır.
- Büyük bir veriden küçük bir özet oluşturulur ve bu özet bilgiden orijinal veriye dönülür.
- Tam güvenli bir algoritma değildir, ama hızlı çalışan bir algoritma olduğu için tercih edilir.
- İstenilen her boyuttaki veriden 128 bitlik bir özet elde eder. Bu özet bilgiyi de 32 bitlik 4 eşit parçaya böler.
- MD5'in sıkıştırma fonksiyonunda bir çakışma bulunmasından yola çıkarak SHA (Secure Hashing Algorithm) gibi algoritmalara yönelinse de MD5'in kırılabilirliği ispat edilmediğinden günümüzde halen veri güvenliği için tercih edilir.
- SHA1 ($2^{64}$-1) bitlik mesajdan 160 bitlik özetleme değeri (hash fonksiyonu) üretir.
- MD5 tek yönlü bir algoritmadır. Geri dönüşümü yoktur. Örneğin : Kullanıcı şifresini unutursa sistem kullanıcıya şifresini veremez. Yeniden şifre oluşturması istenir.
- Eldeki veri 512 bitlik bloklara ayrılır ve her birine aynı işlem uygulanır. Veri 512'in katı değilse padding işlemi uygulanır.